RSAC™ 2025 Conference

Many Voices.
One Community.

SESSION ID: IAIS-T02

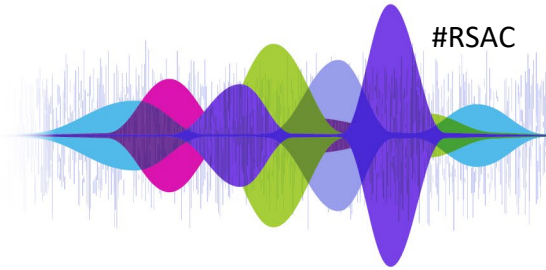
The Future of Secure Programming Using LLMs

Mark Sherman

Technical Director
CERT, Software Engineering Institute
Carnegie Mellon University
mssherman@sei.cmu.edu

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Disclaimer

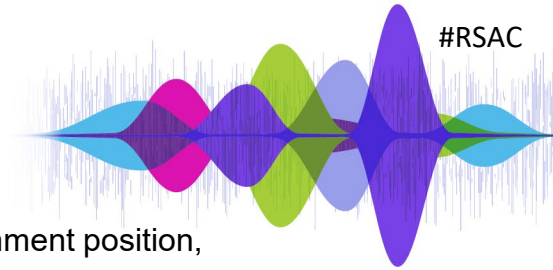


Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference LLC or any other co-sponsors. RSA Conference LLC does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

© 2025 RSA Conference LLC or its affiliates. The RSAC and RSAC CONFERENCE logos and other trademarks are proprietary. All rights reserved.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.



Copyright 2025 Carnegie Mellon University.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific entity, product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute nor of Carnegie Mellon University - Software Engineering Institute by any such named or represented entity.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material is licensed under Creative Commons Attribution-Non-Commercial 4.0 International (CC BY-NC4.0) - https://creativecommons.org/licenses/by-nc/4.0/

Requests for other permissions should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work product was created in part using generative AI.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM25-0250

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Acknowledgements

Robert Schiela
Deputy Technical Director

David Svoboda
Software Security Engineer

David Schulker
Senior Data Scientist

Jamie Glenn
Operations Coordinator

Genavieve Chick
Intern

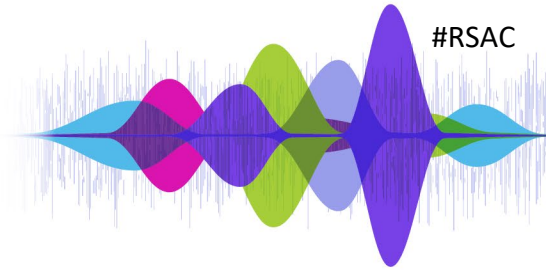
Vedha Avali
Intern

I want to thank the following members of the Cybersecurity Foundations group at CERT for their expert assistance with evaluating and preparing these materials:

- Jamie Glenn — Experiment management
- Robert Schiela — SME
- David Svoboda — SME (Representative, ISO/IEC JTC1/SC22/WG14 (C Programming Language) and lead of Undefined Behaviors Study Group (UBSG))
- David Schulker — Data Scientist and co-Lead
- Genavieve Chick — Data Science Intern
- Vedha Avali — Data Science Intern

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

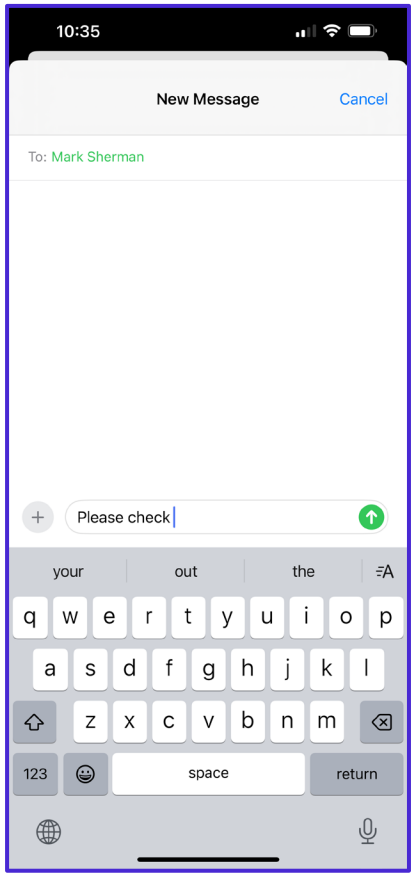
Agenda



- Background on Generative AI
- Technology Expectations
- Initial Technology Evaluations
- Experiment and Results
- Applying the Results
- Summary

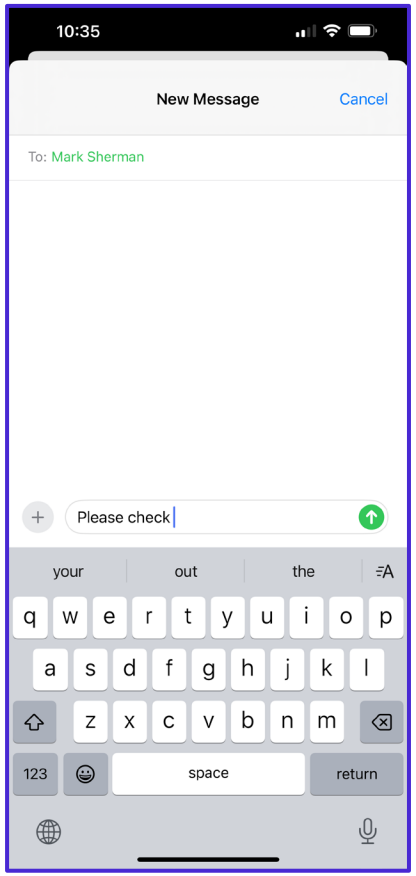
[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Autocomplete is familiar to everyone



[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Autocomplete scaled on steroids



(Image generated by AI at Adobe Images)

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Autocomplete now!

ChatGPT 4o:“Please write a sonnet in the style of Shakespeare about my love of spicy food.”

A Sonnet to Spice

O fiery muse that dost my tongue enflame,
With heat that dances, fierce yet sweetly bold,
No dull repast could ever taste the same,
Once kissed by flames of crimson, bright and gold.

The chili’s burn doth wake my slumb’ring soul,
Like passion’s touch that lingers on my lips,
Its embered fire I willingly extol,
Though scorching pain may bring me to eclipse.

Yet from this blaze I turn not, nor retreat,
For in the heat, my heart doth find delight.
Each molten bite, a battle won in heat,
A war of taste where pleasure meets the fight.

So let the peppers blaze and bring me bliss,
For love is fire, and spice, a burning kiss.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Autocompletion is Established Technology for IDEs Years Ago

Using AI For Autocompletion

```
import os
import sys

# Count lines of code in the given directory, separated by file extension
def main(directory):
    line_count = {}
    for filename in os.listdir(directory):
        _, ext = os.path.splitext(filename)
        if ext not in line_count:
            line_count[ext] = 0
        for line in open(os.path.join(directory, filename)):
            line_count[ext] += 1
            line_count[ext] += 1                13%
            line_count[ext                   Tab  20%
            line_count[ext] +=                3  14%
            line_count[ext].append(           4   3%
            line                              5  23%


```

© 2019 TabNine, See https://tabnine.com/eula

Safe, correct code could be written incrementally

- Using n-grams
- Using deep learning (Generative Pretrained Transformer 2)

Sources:
E. Schutte, Autocomplete from StackOverflow, 2016, https://emilschutte.com/stackoverflow-autocomplete/

(Jacob Jackson) TabNine, "Autocompletion with deep learning," July 18, 2019, https://tabnine.com/blog/deep

L. Tung, "New tool promises to turbo-charge coding in major programming languages," July 25, 2019, https://www.zdnet.com/article/new-tool-promises-to-turbo-charge-coding-in-major-programming-languages/

Mark Sherman, Using AI to Build More Secure Software, Abstractions II Conference, August 23, 2019, https://insights.sei.cmu.edu/documents/4246/2019_017_001_553383.pdf

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Autocomplete now!

ChatGPT 4:“Write a Pascal program that will convert a Julian calendar date to a Hebrew (lunar) calendar date.”

```
program JulianToHebrew;
uses SysUtils, Math;

type
  TDate = record
    Year, Month, Day: Integer;
  end;

var
  JulianDate, HebrewDate: TDate;

function JulianToHebrewDate(Julian: TDate): TDate;
var
  HebrewYear, HebrewMonth, HebrewDay: Integer;
begin
  // This is where the actual conversion algorithm
should be implemented.
  // Placeholder values are assigned for
demonstration.
  HebrewYear := (Julian.Year - 3760); // Simplified
example, adjust accordingly
  HebrewMonth := 1; // Placeholder
  HebrewDay := 1; // Placeholder

  Result.Year := HebrewYear;
  Result.Month := HebrewMonth;
  Result.Day := HebrewDay;
end;
```

```
procedure ReadDate(var Date: TDate);
begin
  Write('Enter year: ');
  ReadLn(Date.Year);
  Write('Enter month: ');
  ReadLn(Date.Month);
  Write('Enter day: ');
  ReadLn(Date.Day);
end;

procedure PrintDate(Date: TDate);
begin
  WriteLn('Year: ', Date.Year, ' Month: ', Date.Month, '
Day: ', Date.Day);
end;

begin
  WriteLn('Enter the Julian date:');
  ReadDate(JulianDate);

  HebrewDate := JulianToHebrewDate(JulianDate);

  WriteLn('Converted Hebrew date:');
  PrintDate(HebrewDate);
end.
```


[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Lots of Hype about Using ChatGPT for Coding

Okay, so ChatGPT just debugged my code. For real.

https://www.zdnet.com/article/okay-so-chatgpt-just-debugged-my-code-for-real/

How to use ChatGPT to write code

By Amber Israelsen | March 22, 2023

https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding

How good is ChatGPT at writing code?

"ChatGPT is a revolutionary AI-based system that can help you generate programming solutions quickly and easily"

https://botpress.com/blog/how-good-is-chatgpt-at-writing-code

Ai

ChatGPT Changed How I Write Software

By Allen Helton __ 31 May 2023

https://www.readysetcloud.io/blog/allen.helton/chatgpt-changed-how-i-write-software/

8 Mar 2023 · Culture

ChatGPT Writes Code: Will It Replace Software Developers?

Written by: Hailey Savona

Reviewed by: Dan Ackerson

"ChatGPT is the solution many in the development community have been looking for."

https://semaphoreci.com/blog/chatgpt-software-developers

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

“Silicon Valley” is Enthusiastic

TECH

Mark Zuckerberg says AI could soon do the work of Meta's midlevel engineers

Lakshmi Varanasi Jan 11, 2025, 1:28 PM ET Share Save

Meta CEO Mark Zuckerberg says the work of midlevel software engineers can soon be outsourced to AI. BRENDAN SMIALOWSKI/ Getty Images

- Mark Zuckerberg said Meta will start automating the work of midlevel software engineers this year.
- Meta may eventually outsource all coding on its apps to AI.

https://www.businessinsider.com/mark-zuckerberg-meta-ai-replace-engineers-coders-joe-rogan-podcast-2025-1

Marc Benioff, Salesforce Founder: Why Salesforce Isn't Hiring Software Engineers | E1236

We have increased the productivity this year … AI … by more than 30%. … we will have less support engineers next year.

https://www.youtube.com/watch?v=Xgsxi7IGMEU
https://www.salesforceben.com/salesforce-will-hire-no-more-software-engineers-in-2025-says-marc-benioff/

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Telling the Tale to Wall Street

Stuart Monk - stock.adobe.com

We need 70% less coders from third parties to code as the AI handles most of the coding, the human only needs to look at the final 30% to validate it.

Murray Auchincloss, CEO & Director
BP Oil
Earnings Call, May 7, 2024

https://seekingalpha.com/article/4690194-bp-p-l-c-bp-q1-2024-earnings-call-transcript

Today, more than a quarter of all new code at Google is generated by AI, then reviewed and accepted by engineers.
Sundar Pichai, CEO Alphabet and Google
Earnings call, Oct 29, 2024

https://abc.xyz/assets/1f/d2/b4b2a1c4437395dce09645f71173/2024-q3-earnings-transcript.pdf

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Academic Research More Nuanced

Participants who had access to an AI assistant wrote significantly less secure code.

Do Users Write More Insecure Code with AI Assistants
Perrry, Srivastava, Kmar, Boneh, CCS '23, Nov 26-20, 2023,
https://arxiv.org/pdf/2211.03622

Copilot [produced] 1,689 programs. Of these, we found approximately 40% to be vulnerable.

Pearce, Ahmad, Tan, Dolan-Gavin, Larri, Asleep at the Keyboard? Assessing the Security of GitHubb CoPilot's Code Contributions, IEEE Symposium on Security and Privacy, 2022, https://arxiv.org/pdf/2108.09293

AI coding tools continue to consistently generate insecure code.

"AI Code, Security and Trust: Organization Must Change Their Approach," 2023 Snyk AI-Generated Code Security Report,
https://go.snyk.io/2023-ai-code-security-report-dwn-typ.html

Researchers from Johannes Gutenberg University and University College London … gave 40 pieces of buggy code to [ChatGPT]. … ChatGPT solved 19 problems.

https://www.pcmag.com/news/watch-out-software-engineers-chatgpt-is-now-finding-fixing-bugs-in-code
https://dl.acm.org/doi/10.1145/3135932.3135941
https://arxiv.org/abs/2301.08653

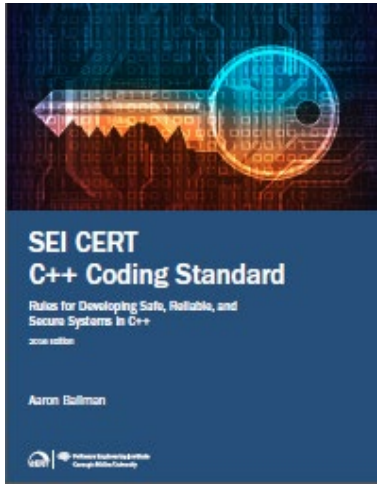
We introduce an evaluation framework consisting of 2,294 software engineering problems … The best-performing model … is able to solve a mere 1.96% of the issues.

Jimenez, Yang, Wettig, Yao, Pei, Press, and Narasimhan, "SWE-BENCH: CAN LANGUAGE MODELS RESOLVE REAL-WORLD GITHUB ISSUES?", Nov 11, 2024,
https://arxiv.org/pdf/2310.06770

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

CERT Secure Coding Standards

- Collected wisdom from thousands of contributors on community wiki since Spring 2006
- http://securecoding.cert.org
  - SEI CERT C Coding Standard
  - SEI CERT C++ Coding Standard
  - CERT Oracle Secure Coding Standard for Java



- Secure Coding Training and Professional Certificates
  - CERT Secure Coding in C and C++
  - CERT Secure Coding in Java
- International Standards Participation
  - ISO/IEC C Programming Language
  - ISO/IEC C++ Programming Language

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Example Standard Page (FIO47-C:Use valid format strings)

Noncompliant Code Example

Mismatches between arguments and conversion specifications may result in undefined behavior. Compilers may diagnose type mismatches in formatted output function invocations. In this noncompliant code example, the `error_type` argument to `printf()` is incorrectly matched with the `s` specifier rather than with the `d` specifier. Likewise, the `error_msg` argument is incorrectly matched with the `d` specifier instead of the `s` specifier. These usages result in undefined behavior. One possible result of this invocation is that `printf()` will interpret the `error_type` argument as a pointer and try to read a string from the address that `error_type` contains, possibly resulting in an access violation.

```
#include <stdio.h>

void func(void) {
  const char *error_msg = "Resource not available to user.";
  int error_type = 3;
  /* ... */
  printf("Error (type %s): %d\n", error_type, error_msg);
  /* ... */
}
```

Compliant Solution

This compliant solution ensures that the arguments to the `printf()` function match their respective conversion specifications:

```
#include <stdio.h>

void func(void) {
  const char *error_msg = "Resource not available to user.";
  int error_type = 3;
  /* ... */
  printf("Error (type %d): %s\n", error_type, error_msg);

  /* ... */
}
```

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Example Standard Page (FIO47-C:Use valid format strings)

Noncompliant Code Example

Mismatches between arguments and conversion specifications may result in undefined behavior. Compilers may diagnose type mismatches in formatted output function invocations. In this noncompliant code example, the `error_type` argument to `printf()` is incorrectly matched with the `s` specifier rather than with the `d` specifier. Likewise, the `error_msg` argument is incorrectly matched with the `d` specifier instead of the `s` specifier. These usages result in undefined behavior. One possible result of this invocation is that `printf()` will interpret the `error_type` argument as a pointer and try to read a string from the address that `error_type` contains, possibly resulting in an access violation.

```
#include <stdio.h>

void func(void) {
  const char *error_msg = "Resource not available to user.";
  int error_type = 3;
  /* ... */
  printf("Error [type %s): %d\n", error_type, error_msg);
  /* ... */
}
```

(type %s): %d\n

Compliant Solution

This compliant solution ensures that the arguments to the `printf()` function match their respective conversion specifications:

```
#include <stdio.h>

void func(void) {
  const char *error_msg = "Resource not available to user.";
  int error_type = 3;
  /* ... */
  printf("Error [type %d): %s\n", error_type, error_msg);

  /* ... */
}
```

(type %d): %s\n

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

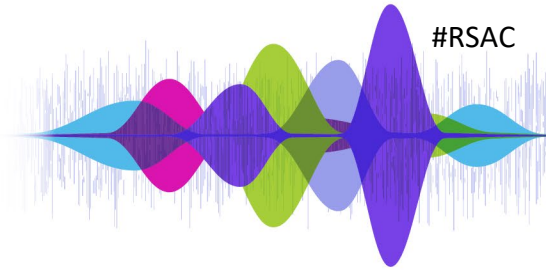
Experiments – What is the State of the Practice and is It Improving?

- Ask ChatGPT to identify the presence or absence of errors in 1223 examples of compliant and noncompliant code from SEI CERT C, C++ and Java Secure Coding Standards
  - ChatGPT 3.5 as of March 27, 2023, and Jun-Aug 2024
  - ChatGPT 4 and GitHub CoPilot as of February 16, 2024
  - ChatGPT 4o as of Jun-Aug 2024
  - 2684 runs, each trial run as a new conversation
  - No trial systematically repeated
  - All examples (with solutions) available on the web during ChatGPT's training
  - Some examples have comments suggesting error
  - ChatGPT's performance evaluated by SMEs

Mark Sherman, "Using ChatGPT to Analyze Your Code? Not So Fast," February 12, 2024, https://insights.sei.cmu.edu/blog/using-chatgpt-to-analyze-your-code-not-so-fast/
David Schulker, "GenAI for Code Review of C++ and Java," November 18, 2024 https://insights.sei.cmu.edu/blog/genai-for-code-review-of-c-and-java/

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Finding and fixing an error (FIO47-C:Use valid format strings) ChatGPT 3.5



Example of noncompliant code:

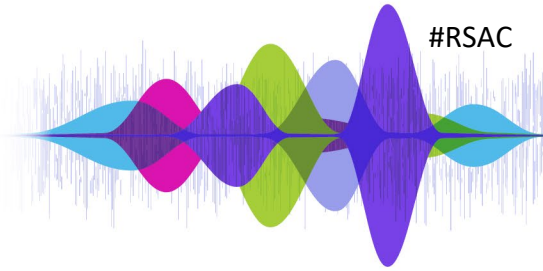
```
#include <stdio.h>


void func(void) {
  const char *error_msg = "Resource not available to user.";
  int error_type = 3;
  /* ... */
  printf("Error (type %s): %d\n", error_type, error_msg);
  /* ... */
}
```

ChatGPT3.5:

The format specifier in the printf statement is incorrect. %s is used for printing strings, but error_type is an integer. To print an integer, you should use %d instead. Similarly, error_msg is a string, so you should use %s to print it.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Finding and fixing an error (FIO47-C:Use valid format strings) ChatGPT 4.0 and CoPilot



ChatGPT 4.0

The code snippet you've shared contains a mistake in the `printf` function call. The format specifiers in the `printf` function are not correctly matched to the types of the variables being printed. Specifically, the format specifier `%s` is used for strings (character arrays), and `%d` is used for integers. However, in your `printf` call, the specifiers are incorrectly matched to the variables `error_type` and `error_msg`.

The correct `printf` statement should be as follows:

printf("Error (type %d): %s\n", error_type, error_msg);

Here, `%d` correctly corresponds to `error_type` (an integer), and `%s` correctly corresponds to `error_msg` (a string). This change correctly matches the format specifiers to the types of the variables being printed.

CoPilot

The issue with your code is in the printf function call. You are trying to print an integer (error_type) with the %s format specifier, which is for strings, and a string (error_msg) with the %d format specifier, which is for integers.

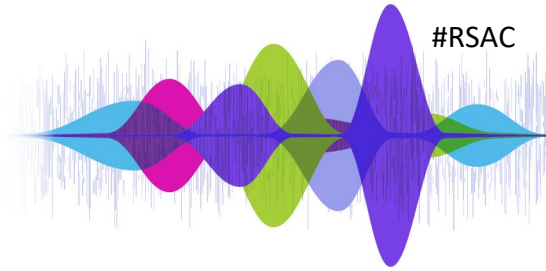
Here is the corrected code:

printf("Error (type %d): %s\n", error_type, error_msg);

This will correctly print the error_type as an integer and error_msg as a string.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

All Versions Missing Errors



DCL38-C Use the correct syntax when declaring a flexible array member

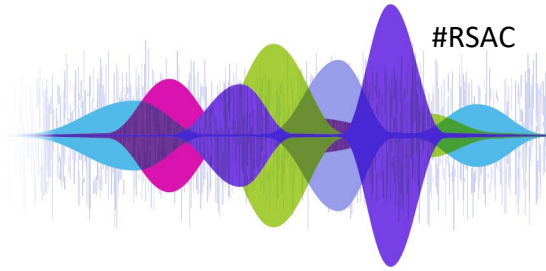
ChatGPT3.5: There is nothing technically wrong with this code

ChatGPT4: the provided code snippet is an appropriate use of a flexible array member within a struct for dynamic array allocation in C.

CoPilot: The code seems to be correct in terms of syntax and logic.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Improved Technology Sometimes Gives Better Results



EXP33-C Do not read uninitialized memory

ChatGPT3.5: The code appears to be incomplete as it doesn't have any obvious errors.

ChatGPT4: There are a couple of issues with the provided code … **Uninitialized Variable:**

CoPilot: The issue with this code is that the sign variable in the is_negative function is not initialized.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Memorization – reflecting code back

IDS00-J. Prevent SQL injection (Java)

Example of compliant code:

```
public void doPrivilegedAction(
  String username, char[] password
) throws SQLException {
  Connection connection = getConnection();
  if (connection == null) {
    // Handle error
  }
  try {
    String pwd = hashPassword(password);

    // Validate username length
    if (username.length() > 8) {
      // Handle error
    }

    String sqlString =
      "select * from db_user where username=? and password=?";
    PreparedStatement stmt = connection.prepareStatement(sqlString);
    stmt.setString(1, username);
    stmt.setString(2, pwd);
    ResultSet rs = stmt.executeQuery();
    if (!rs.next()) {
      throw new SecurityException("User name or password incorrect");
    }

    // Authenticated; proceed
  } finally {
    try {
      connection.close();
    } catch (SQLException x) {
      // Forward to handler
    }
  }
}
```

ChatGPT3.5:

The code does not securely handle passwords because it is converting the character array representing the password directly into a String before hashing it. Converting the password to a String can leave it vulnerable to exposure in memory since strings are immutable and will remain in memory until they are garbage collected. To address this issue, the password should be hashed directly from the character array without converting it into a String.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Memorization – reflecting code back

IDS00-J. Prevent SQL injection (Java)

Example of compliant code:

```
public void doPrivilegedAction(
  String username, char[] password
) throws SQLException {
  Connection connection = getConnection();
  if (connection == null) {
    // Handle error
  }
  try {
    String pwd = hashPassword(password);

    // Validate username length
    if (username.length() > 8) {
      // Handle error
    }

    String sqlString =
      "select * from db_user where username=? and password=?";
    PreparedStatement stmt = connection.prepareStatement(sqlString);
    stmt.setString(1, username);
    stmt.setString(2, pwd);
    ResultSet rs = stmt.executeQuery();
    if (!rs.next()) {
      throw new SecurityException("User name or password incorrect");
    }

    // Authenticated; proceed
  } finally {
    try {
      connection.close();
    } catch (SQLException x) {
      // Forward to handler
    }
  }
}
```

String pwd = hashPassword(password);

ChatGPT3.5:

```
ection();

  }
  try {
    // Hash the password directly from the char array
    String pwd = hashPassword(password);

    // Validate username length
    if (username.length() > 8) {
      // Handle error
    }

    String sqlString = "select * from db_user where username=? and password=?";
    PreparedStatement stmt = connection.prepareStatement(sqlString);
    stmt.setString(1, username);
    stmt.setString(2, pwd);
    ResultSet rs = stmt.executeQuery();
    if (!rs.next()) {
      throw new SecurityException("User name or password incorrect");
    }

    // Authenticated; proceed
  } finally {
    try {
      connection.close();
    } catch (SQLException x) {
      // Forward to handler
    }
  }
}
```


[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Memorization – reflecting code back

IDS00-J. Prevent SQL injection (Java)

Example of compliant code:

```
public void doPrivilegedAction(
  String username, char[] password
) throws SQLException {
  Connection connection = getConnection();
  if (connection == null) {
    // Handle error
  }
  try {
    String pwd = hashPassword(password);

    // Validate username length
    if (username.length() > 8) {
      // Handle error
    }

    String sqlString =
      "select * from db_user where username=? and password=?";
    PreparedStatement stmt = connection.prepareStatement(sqlString);
    stmt.setString(1, username);
    stmt.setString(2, pwd);
    ResultSet rs = stmt.executeQuery();
    if (!rs.next()) {
      throw new SecurityException("User nam
    }

    // Authenticated; proceed
  } finally {
    try {
      connection.close();
    } catch (SQLException x) {
      // Forward to handler
    }
  }
}
```

String pwd = hashPassword(password);

ChatGPT3.5:

```
ection();

  }
  try {
    // Hash the password directly from the char array
    String pwd = hashPassword(password);

    // Validate username length
    if (username.length() > 8) {
      // Handle error
    }

    String sqlString = "select * from db_user where username=? and password=?";
                       n.prepareStatement(sqlString);

                                  name or password incorrect");

  } finally {
    try {
      connection.close();
    } catch (SQLException x) {
      // Forward to handler
    }
  }
}
```

4o also complained about password hashing and the (nonexistent) SQL Injection error in the compliant version

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Evaluating Correction Can be Difficult EXP42-C Standard

Noncompliant Example

```
#include <string.h>

struct s {
  char c;
  int i;
  char buffer[13];
};

void compare(const struct s *left, const struct s *right) {
  if ((left && right) &&
      (0 == memcmp(left, right, sizeof(struct s)))) {
    /* ... */
  }
}
```

0 == memcmp(left->buffer, right->buffer, 13)

char buffer[13]

Compliant recommendation

```
  int i;
  char buffer[13];
};

void compare(const struct s *left, const struct s *right) {
  if ((left && right) &&
      (left->c == right->c) &&
      (left->i == right->i) &&
      (0 == memcmp(left->buffer, right->buffer, 13))) {
    /* ... */
  }
}
```


[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

ChatGPT 3.5's Subtle Mistaken Fix

Real problem

The code appears to be comparing two instances of a struct **s** using **memcmp()**.

However, this may not always produce the expected results, especially if the struct contains padding bytes between its members. The padding bytes may have different values in the two instances of the struct, which would cause **memcmp()** to return a non-zero value even if the actual data members are the same.

0 == strcmp(left->buffer, right->buffer)

To avoid this issue, it is recom
example, the code could be modified as follows:

```
void compare(const struct s *left, const struct s *right) {
  if ((left && right) &&
      (left->c == right->c) &&
      (left->i == right->i) &&
      (0 == strcmp(left->buffer, right->buffer))) {
    /* ... */
  }
}
```

Here, the struct members are compared individually using **==** for **char** and **int** members and **strcmp()** for the **buffer** member.

- Assumed "buffer" is a string
- If string, should be using strncmp, not strcmp

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

ChatGPT 4 Did Better

```
int are_structs_equal(const struct s *left, const struct s *right) {
  if (left == right) {
    // They are the same instance or both are NULL.
    re
  }
  if (!left || !right) {
    // One is NULL and the other is not.
    return 0;
  }
  // Compare each member individually.
  return (left->c == right->c) &&
         (left->i == right->i) &&
         (0 == memcmp(left->buffer, right->buffer, sizeof(left->buffer)));
}
```

0 == memcmp(left->buffer, right->buffer, sizeof(left->buffer))

"The code snippet you've provided defines a struct `s` and a function `compare` that compares two instances of `s` for equality using `memcmp`. While at first glance, the `compare` function might seem correct, there are a couple of issues related to the use of `memcmp` for comparing structs in C that need to be considered:

1. **Padding bytes**: The C standard allows compilers to insert padding bytes

initialized), `memcmp` may report that two structs are different, even if all their actual member variables are identical.

2. **Floating point numbers**: If your struct contained floating point numbers, comparing them with `memcmp` could lead to incorrect results due to the way floating point numbers are represented in memory. This is not directly relevant to your current struct but is something to keep in mind when using `memcmp` for comparison.

3. **Pointers**: If the struct contained pointer members, `memcmp` would only compare the pointer values (addresses) rather than the data they point to. This is not an issue in your current struct but is another consideration when using `memcmp`.

To work around the padding byte issue, you might need to compare each member individually, especially if there's a possibility of padding bytes between members. For the current struct `s`, comparing each member individually would look something like this:"

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

CoPilot Did Better

```
int compare(const struct s *left, const struct s *right) {
  if (left == NULL || right == NULL) {
    return 0; // or handle error as you see fit
  }
  if (left->c != right->c) {
    retu
  }
  if (left->i != right->i) {
    return 0;
  }
  if (memcmp(left->buffer, right->buffer, sizeof(left->buffer)) != 0) {
    return 0;
  }
  return 1;
}
```

memcmp(left->buffer, right->buffer, sizeof(left->buffer)) != 0

“The problem with this code is that it uses `memcmp` to compare two structures. This is not safe because the padding bytes (which are not members could be different, leading to a false negative result even if the meaningful fields are identical.”

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Identifies a non-existent problem and misses real problem

(ENV33-C. Do not call system())

In this noncompliant code example, the `system()` function is used to execute `any_cmd` in the host environment.

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

enum { BUFFERSIZE = 512 };

void func(const char *input) {
  char cmdbuf[BUFFERSIZE];
  int len_wanted = snprintf(cmdbuf, BUFFERSIZE,
                            "any_cmd '%s'", input);
  if (len_wanted >= BUFFERSIZE) {
    /* Handle error */
  } else if (len_wanted < 0) {
    /* Handle error */
  } else if (system(cmdbuf) == -1) {
    /* Handle error */
  }
}
```

snprintf(…

system(cmdbuf)

ChatPGT3.5:

it's recommended to **use** a function that is specifically designed to handle string concatenation and avoid buffer overflows, such as **snprintf().**

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Revisiting ENV33 – Better

ChatGPT4:

Security Vulnerability (Command Injection): The most critical issue with this code is that it directly incorporates user input (`input`) into a command string that is executed by the system.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Revisiting ENV33 – Better – and surprising guardrails

ChatGPT4:

Security Vulnerability (Command Injection): The most critical issue with this code is that it directly incorporates user input (`input`) into a command string that is executed by the system.

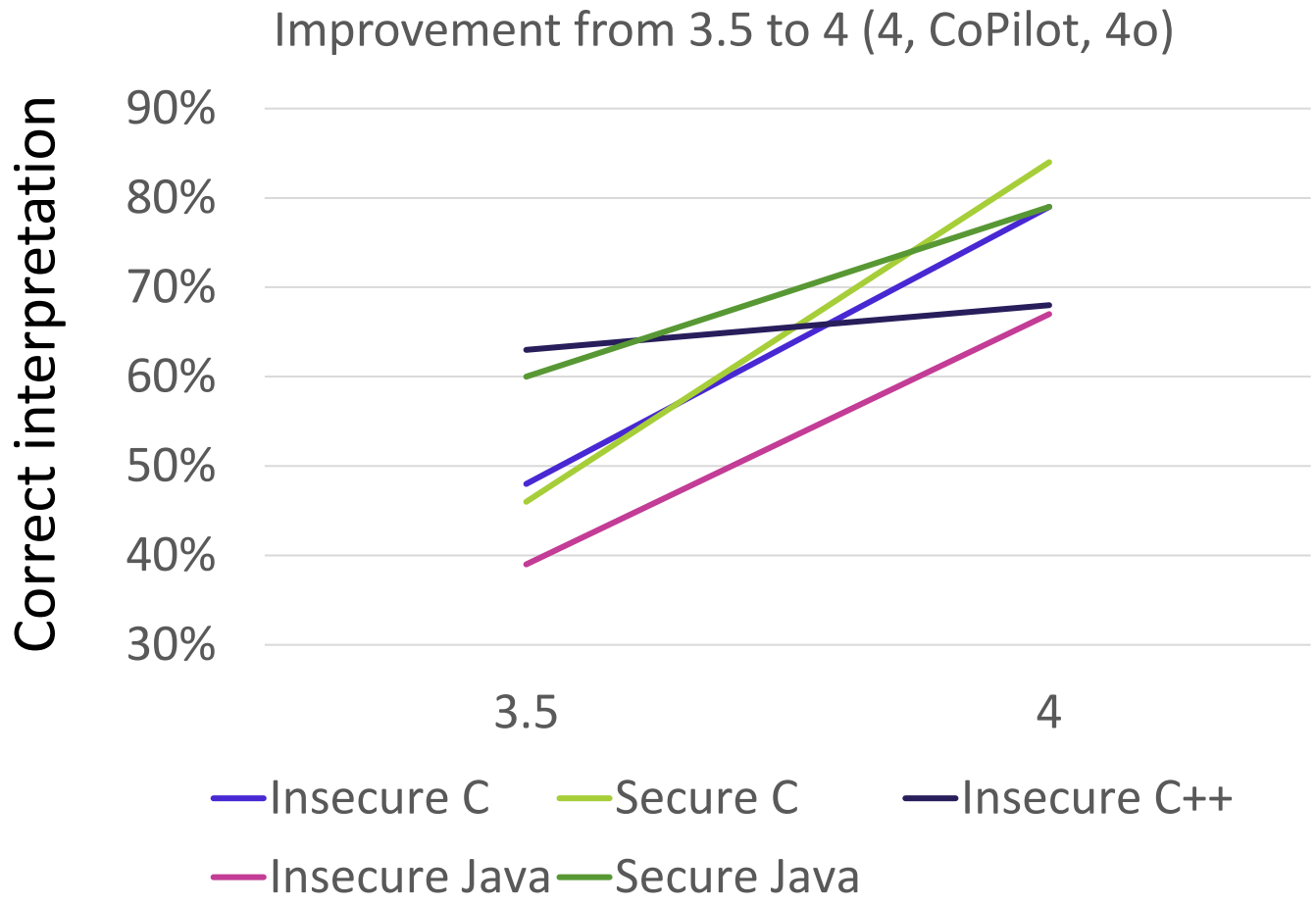
CoPilot:

Sorry, the response was filtered by the Responsible AI Service. Please rephrase your prompt and try again. Learn more.

(Tried 3 times.)

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Improved performance



[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

What Areas Improved?

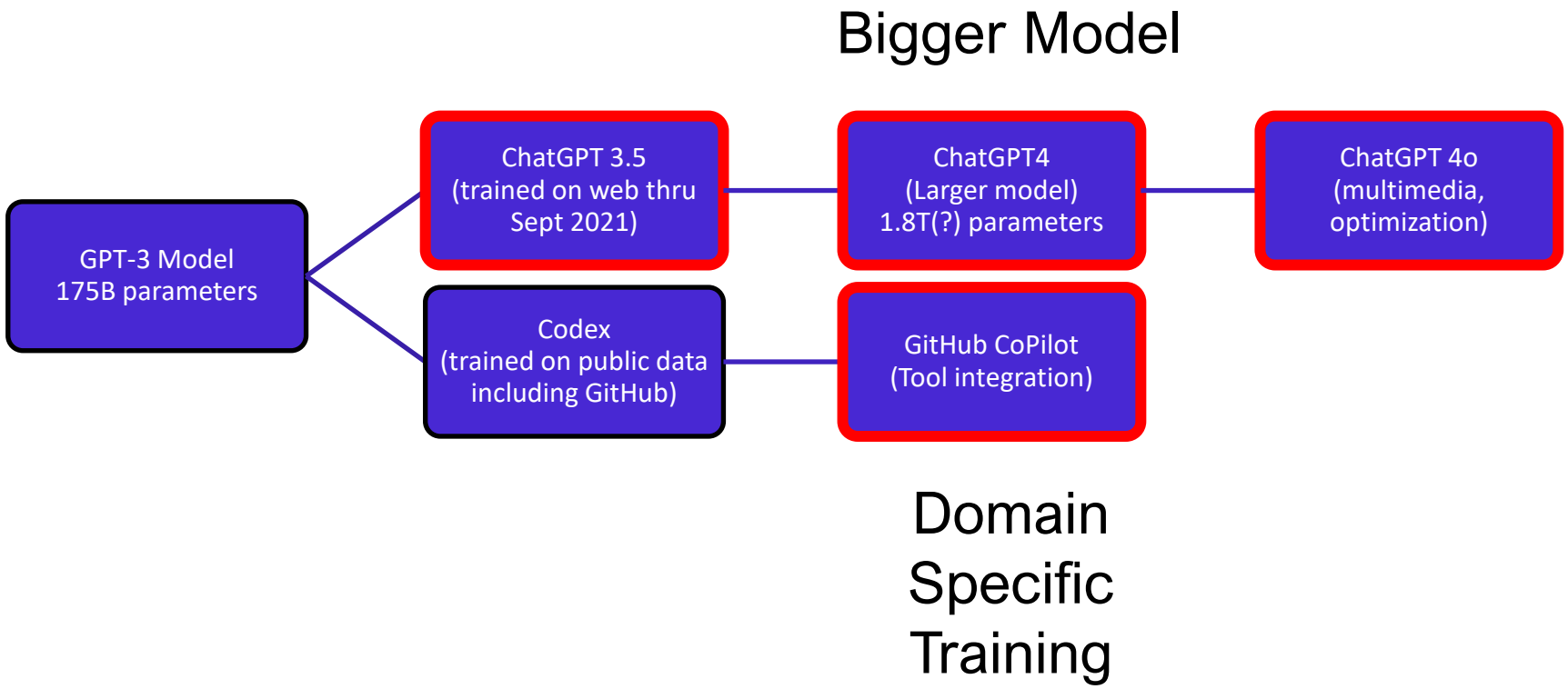
Noncompliant C examples: Common mistakes getting good coverage.

Focused on what is trained on

	Total tests	3.5 Found error	4/C Improve-ments over 3.5	4/c + 3.5 Found error	% Improve ment	% Effective ness
PRE	4	1	1	2	100%	50%
DCL	21	11	1	12	9%	57%
EXP	**49**	**33**	**14**	**47**	**42%**	**96%**
INT	25	11	6	17	55%	68%
FLP	10	2	4	6	200%	60%
ARR	20	7	8	15	114%	75%
STR	**19**	**12**	**4**	**16**	**33%**	**84%**
MEM	**13**	**8**	**4**	**12**	**50%**	**92%**
FIO	21	3	6	9	200%	43%
ENV	9	4	1	5	25%	56%
SIG	6	1	0	1	0%	17%
ERR	12	5	4	9	80%	75%
CON	17	7	6	13	86%	76%
MSC	12	5	3	8	60%	67%

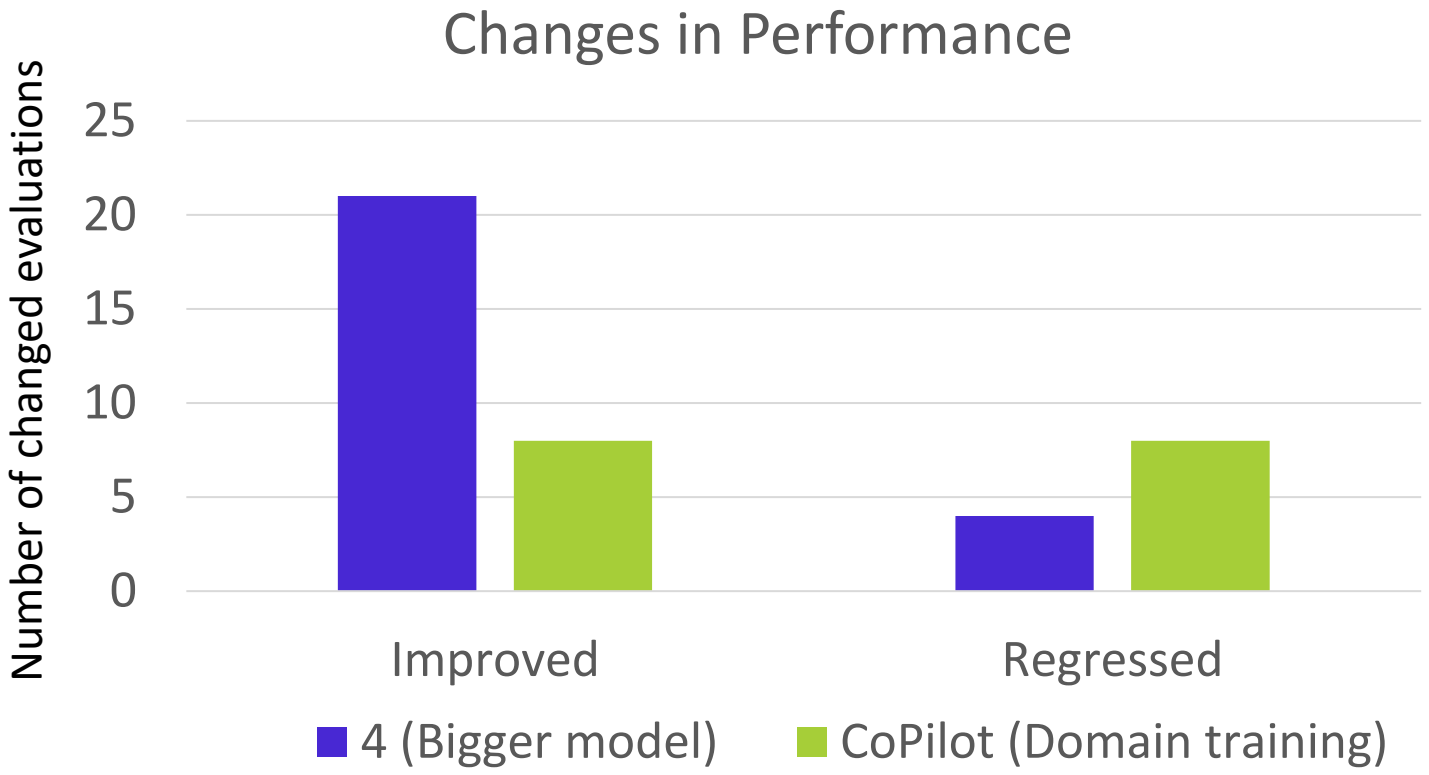
[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Systems used in experiments



[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

An Unexpected Result: General vs Domain Training (Insecure C)



[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

General vs Specially Trained LLMs – General Software

General purpose models

Code trained models

Table 3. Performance comparison of LLMs on HumanEval.

Type	Model	Pass@1 on HumanEval
Closed-source	GPT-4	83.5 (1)
	GPT-3.5	76.8 (2)
Open-source	CodeLLaMa-7B	39.6 (8)
	CodeLLaMa-13B	42.7 (7)
	CodeLLaMa-34B	52.4 (3)
	DeepSeekCoder-6.7B	45.1 (5)
	DeepSeekCoder-33B	50.6 (4)
	StarCoder	34.8 (9)
	StarCoder2-3B	31.1 (11)
	StarCoder2-7B	34.8 (9)
	StarCoder2-15B	45.1 (5)

ZHENG, WANG, SHI, ZHANG, & ZHENG, 'How Well Do LLMs Generate Code for Different Application Domains? Benchmark and Evaluation," Dec 2024, https://arxiv.org/pdf/2412.18573DEWU

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Field Experience – Consultant and Developer

"If you're not a great coder, you're not going to be able to have the judgement to know if the code that comes out is going to do what you want in the way you want.

Bret Greenstein, Data and AI Leader, PwC,

Can AI coding Tools meet enterprise expectations, (quoted by Lindsey Wilkinson, Feb 26. 2024, https://www.ciodive.com/news/generative-ai-coding-tools-reality-versus-hype/708426/

"When LLMs make mistakes — and they do, regularly — it takes an experienced developer to spot these issues, understand their implications, and know how to fix them."

Rahul Agarwal, Machine Learning Engineer, Facebook,

LLMs Won't "Why Replace Programmers: A Reality Check," Nov 25, 2024, https://mlwhiz.medium.com/why-llms-wont-replace-programmers-a-reality-check-73133bd3b9cc

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Finding Correct Code is Fatiguing

"Recent work shows that 93% of developers are only willing to review up to ten patches."

Jiang, Liu, Lutellier, Tan, "Impact of Code Language Models on Automated Program Repair," 2023 IEEE/ACM 45rd International Conference on Software Engineering (ICSE)

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Users Defer to Perceived AI Authority

Image generated by OpenAI

"**Participants with access to an AI assistant were also more likely to believe they wrote secure code**, suggesting that such tools may lead users to be overconfident about security flaws in their code."

Do Users Write More Insecure Code with AI Assistants

Perrry, Srivastava, Kmar, Boneh, CCS '23, Nov 26-20, 2023, https://arxiv.org/pdf/2211.03622

"Respondents commonly found security issues with AI suggestions but they also **strongly believed that AI suggestions were secure**."

"AI Code, Security and Trust: Organization Must Change Their Approach," 2023 Snyk AI-Generated Code Security Report, https://go.snyk.io/2023-ai-code-security-report-dwn-typ.html,

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Controlling the Narrative is Key

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Prompting Improves Results

Watch Out, Software Engineers: ChatGPT Is Now Finding, Fixing Bugs in Code

By Emily Dreibelbis
January 27, 2023

Courtesy of computer science researchers from Johannes Gutenberg University and University College London …

Researchers gave 40 pieces of buggy code to four different code-fixing systems … ChatGPT solved 19 problems, …. **Providing … hints to ChatGPT, its success rate … increased, fixing 31 out of 40 bugs.**

https://www.pcmag.com/news/watch-out-software-engineers-chatgpt-is-now-finding-fixing-bugs-in-code

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Giving More Guidance Improves Performance

Prompt 0: No guidance
Prompt 1: Generate fix for CWE-xxx
Prompt 2: Include static analyzer alerts
Prompt 3: Add context of error
Prompt 4: Describe repair task

Adapted from Figure 2, Liu, Wang, Zheng, Zhang, "Prompt Fix: Vulnerability Automatic Repair Technology Based on Prompt Engineering," 2024 Workshop on Computing Networking and Communications, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10556123

Prompt Level	% Fixed (low)	% Fixed (high)
0	20	30
1	29	40
2	45	60
3	72	90
4	45	50

Prompt Level	Programs examined	Error Found	% Found
No Prompt	852	646	76%
Extensive Prompt	1015	988	97%

Extracted and adapted from Table 1, Vulnerable C/C++ Programs; Zhang, Liu, Zeng, Yang, K. Li, Y. Li, "Prompt-Enhanced Software Vulnerability Detection Using ChatGPT," 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), https://dl.acm.org/doi/pdf/10.1145/3639478.3643065

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

More Direction Gives Better Results

- Which libraries to include
- Which APIs to use
- Functional documentation
- Target function size ~16 LOC
- Limited to a domain & train model on that specific domain

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

General vs Specially Trained LLMs – Domain Specific Software

General purpose models

Code trained models

Table 4. Performance of LLMs on Various Domains. Abbreviations: **BC**: Blockchain, **CS**: Cloud Service, **DA**: Data Analysis, **DL**: Deep Learning, **DApp**: Desktop Application, **DSys**: Distributed System, **EApp**: Enterprise Application, **Game**: Game Development, **IoT**: Internet of Things, **Mob**: Mobile Application, **Rob**: Robotics, **Web**: Web Development.

LLMs	BC	CS	DA	DL	DApp	DSys	EApp	Game	IoT	Mob	Rob	Web
GPT-4	46.83	39.10	39.02	36.47	43.15	40.58	36.40	41.13	39.46	46.33	42.26	34.45
GPT-3.5	32.19	32.46	29.02	29.72	23.49	20.74	19.26	19.02	22.42	27.09	25.07	17.27
CodeLLaMa-7B	43.24	39.28	33.34	33.34	40.52	38.47	35.76	35.50	38.20	42.78	40.32	34.58
CodeLLaMa-13B	47.91	41.47	36.04	35.69	42.80	40.11	37.05	35.66	40.64	44.39	42.84	35.33
CodeLLaMa-34B	48.90	42.57	37.80	38.25	43.31	42.14	37.80	35.38	41.09	46.07	43.56	36.80
DeepSeekCoder-6.7B	49.04	41.60	36.03	35.66	43.79	40.51	37.22	34.13	40.69	42.03	44.13	34.93
DeepSeekCoder-33B	52.75	44.32	39.65	39.89	46.29	42.27	39.64	39.71	43.45	44.33	48.54	38.88
StarCoder-15.5B	40.17	39.02	34.10	37.19	37.37	35.32	32.84	35.91	37.17	30.45	35.97	30.41
StarCoder2-3B	40.21	37.60	31.97	33.92	35.16	32.38	30.87	34.01	35.31	31.91	36.74	29.09
StarCoder2-7B	41.20	39.57	34.10	35.22	38.03	33.49	33.32	36.43	37.10	34.70	37.81	30.60
StarCoder2-15B	41.67	41.92	37.06	37.67	39.33	36.71	34.87	36.00	39.79	34.53	40.82	33.83

ZHENG, WANG, SHI, ZHANG, & ZHENG, 'How Well Do LLMs Generate Code for Different Application Domains? Benchmark and Evaluation," Dec 2024, https://arxiv.org/pdf/2412.18573DEWU

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Amazon's Experience

Amazon Experience using LLMs for Software Development

Task: Upgrade existing Java 8 and Java 11 programs to Java 17

Claim: saved the equivalent of 4,500 developer-years

Approach

- Train an LLM specifically on Java 8/11 to Java 17 upgrade
- Process programs written according to Amazon standards
- Leverage OpenRewrite recipes for describing the changes
- Reuse existing unit tests for existing programs to validate successful conversion

Jassy, A., Sept 2024, https://www.linkedin.com/posts/andy-jassy-8b1615_one-of-the-most-tedious-but-critical-tasks-activity-7232374162185461760-AdSz/

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Recommendation: Keep Teaching the Model

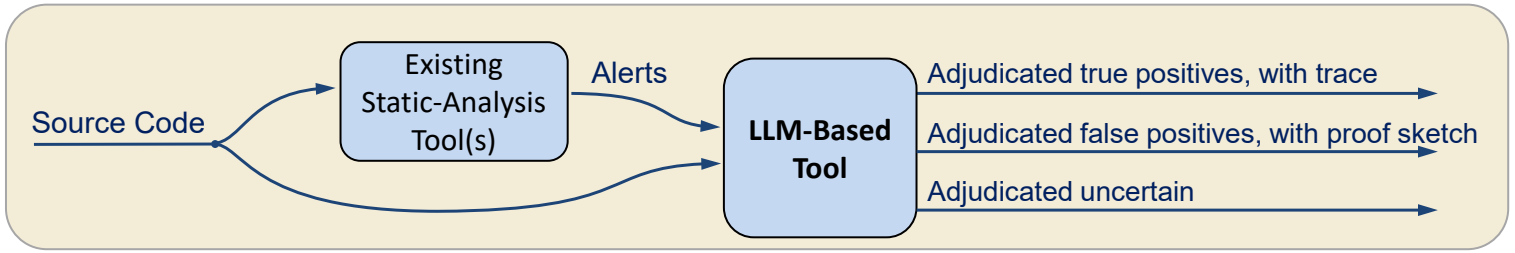
"One of the common criticisms of Large Language Models is that they often output wrong code. Which is true … You want to have a decent output for ChatGPT? Just copy paste a ton of example code upfront. Copy paste your class definitions, the comments to your class definition, maybe some DDL, some example CSV. Paste it often, paste it every time it diverges. Take its code, correct it, paste it back. Paste the documentation page. Paste the entire StackOverflow thread. Paste paste paste, context context context is what is needed here."

Manuel Odendahl, Developer, Mento

Manuel Odendahl , "LLMs will fundamentally change software engineering," Posted on Mar 23, 2023, Edited on Mar 26, 2023 , https://dev.to/wesen/llms-will-fundamentally-change-software-engineering-3oj8

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Drive LLM to Validate Itself



- Use other tools to generate candidate issues
- Use prompt engineering to focus on candidate issues
- Use LLM to generate patches
- Use LLM to generate evidence of patch correctness
- Validate patch correctness externally, e.g., SMT solvers (aka theorem prover)

Figure 1. Using LLMs for SA Alert Adjudications. Klieber, Flynn, “Using LLMs for SA alert adjudication and rationales”, CrossTalk, 2024

Klieber and Flynn, “Using LLMs to Adjudicate Static-Analysis Alerts,” 58th Hawaii International Conference on System Sciences (HICSS), January 10, 2025, “https://scholarspace.manoa.hawaii.edu/items/42b220fa-17f8-4cfe-9f2b-ba4553e21bc3

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Summary

- Experiments illustrate promise but also limitations
- Like many applications of LLMs, knowledgeable users must review output
- Unfortunately, programmers are challenged to read and evaluate a lot of LLM output; technology assistance is necessary and being developed
- Technology is improving, can be applied when managing expectations

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Apply

- Don’t abandon existing methods for improving software security
  - Secure coding practices and source code analysis tools remain valuable
- Educate management (and developers) on practical applications of LLMs
  - Separate hype from capability
- Adopt low hanging fruit, e.g., autocompletion, documentation surfing, contained code “snippets”
  - Easy productivity gains from basic function
  - Correctness and security manifest
- If situation allows, consider investing in specialty applications
  - Extremely well-defined context and target
  - Have or acquire technical talent to construct LLM applications
  - Practical method for evaluation of LLM output

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

RSAC™ 2025 Conference

Many Voices.
One Community.

Mark Sherman
Carnegie Mellon University
Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
412-268-5800, 888-201-4479
info@sei.cmu.edu

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.